Managing OpenBiblio Plugins

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

OpenBiblio has a very flexible plugin system. This document will describe how you can enable and disable plugins, understand the plugins that are included with OpenBiblio, and design a new plugin.

1 Enabling plugins

Tip
You will need access to the Tools menu.

1. Go to Tools → Plugin Manager.
2. Make sure that Plugins are Allowed.
3. Check the box next to each plugin you are interested in using.

2 Plugins bundled with OpenBiblio

These plugins all come bundled with a default OpenBiblio installation.

3 CSS Utilities (cssUtils)

This plug-in will looks for two categories of CSS issues: 1. CSS entries in . . . /styles.css that appear to not be in use 2. CSS classes referenced in OB that do not appear in . . . /styles.css

Warning
No fixes are suggested or made. The user should be VERY careful in removing any of these without a thorough test.

4 Call Number Utilities (CallNoUtils)

This

4.1 Find Records without call numbers

```
As stated, this function will scan the entire database and return a list of all ↩
   biblios which do not have a call number assigned.
At this time there is no mechanism to bring up the offending record for editing.
```

4.2 Dry Run of call number add

This section describes the dry run feature.

4.3 Add call numbers to records

This section describes the add feature.

5 Creating a new plugin

This requires basic PHP skills and access to the files.

5.1 Design the plugin

1. Create a new directory that begins with the word plugin, followed by an underscore, followed by the name of your plugin. In UNIX systems, you might type something like this:

```
mkdir plugin_excitingPlugin
```

2. If you need to include custom CSS or JS, add it to a file called custom_head.php in this new directory.
3. Create three PHP files in the directory: a JS file, a Srvr file, and a Forms file.

5.1.1 Create a plugin server

Your plugin's server — located at `plugin_excitingPlugin/excitingSrvr.php` — is responsible for three tasks:

1. Performing any actions the user requests.
2. Obtaining any relevant data from the database, file structure, or external API.
3. Displaying the relevant data to the user in a nice HTML format.

A basic plugin server consists of a require statement to get the necessary data and methods and a switch statement to take care of the `$_REQUEST['mode']` variable. Here's a simple example:

```
<?php
        require_once('../shared/common.php');
        switch ($_REQUEST['mode']){
                case 'exciting':
                        echo 'This plugin is very exciting.';
                        break;
                case 'boring':
                        echo 'This plugin is pretty boring.';
                        break;
                default:
                        echo T('invalid mode');
                        break;
        }
```

If you would like to access data from the database, you should require the appropriate model from the model directory. For example, if you'd like to use the Biblios model, you would include this in your plugin_excitingPlugin/excitingSrvr.php:

```
require_once('../model/Biblios.php');
$bib = new Biblios;
```

You can then access all those great methods and data through the `$bib` object.

5.2 Add your plugin to the OpenBiblio menu

1. Create a new directory inside the plugins directory with the name of your plugin. In UNIX systems, you might type something like this:

```
mkdir plugins/excitingPlugin
```

2. Create a file within this new directory called nav.nav

```
<?php
Nav::node('tools/excitingPlugin', 'Exciting Plugin', '../plugin_excitingPlugin/ ↩
    excitingPluginForms.php');
```

3. If you'd like to use Obib's localization framework but need to localize some more terms, create a file called tran.tran. You can use this to add new terms to your locale.

> **Tip**
> Plugins already have all of the exciting localization ability built in. However, note that you cannot override localized term definitions within tran.tran.

+ WARNING: There is not yet a place to specify localized terms for plugins.

4. Turn on your plugin using Tools → Plugin Manager.